Санкт-Петербургский государственный университет
Факультет Прикладной математики - Процессов управления

Кафедра Технологии программирования

Коробков Никита Александрович

Выпускная квалификационная работа бакалавра

Метод межъязыковой адаптации диалоговых систем

Направление 01.03.02
Прикладная математика и информатика

Научный руководитель:
старший преподаватель Мишенин А.Н.

Санкт-Петербург
2019

Оглавление

Введение **2**

Постановка задачи **5**

Обзор литературы **7**

1. Обработка данных **10**

2. Алгоритм **13**

2.1. Статистический анализ 13

2.1.1. Метод 13

2.1.2. Анализ 15

2.2. Модели векторов предложений 16

2.2.1. FastText – среднее 17

2.2.2. FastText – взвешенное среднее 17

2.2.3. ELMo – среднее 18

2.2.4. Universal Sentence Encoder 19

2.3. Линейное преобразование 20

2.4. Нейронная сеть 21

3. Результаты **23**

3.1. Модели векторов предложений 23

3.2. Линейное преобразование 23

3.3. Нейронная сеть 24

Выводы **27**

Заключение **28**

Список литературы **29**

Введение

В последние годы все больше внимания уделяется решению различных задач с применением машинного обучения и нейронных сетей. Одним из наиболее интересных направлений в области применения современных нейросетевых методов являются диалоговые системы.

Диалоговая система – это набор программ и алгоритмов, позволяющих человеку вести диалог с программой в манере, свойственной человеку. Иногда диалоговые системы называют разговорным искусственным интеллектом или "чат-ботом".

Задача построения подобных программ является актуальной для промышленной области, потому что решение задач пользователей путем диалога с агентом поддержки всегда было и будет оставаться наиболее простым и эффективным методом. Так как вербальное общение – наиболее естественный для человека способ коммуникации. Построение хорошей и надежной диалоговой системы заменяющей работников колл центра, позволило бы существенно сократить затраты компаний на найм персонала.

Диалоговые системы можно разделить на системы общего назначения и задачеориентированные. Чем более узкую задачу призвана решать система, тем проще она может быть устроена. В самом тривиальном случае система может просто возвращать заранее известный ответ. Например, текущее время.

Мы хотели бы построить систему, решающую более общую задачу. Для этого в нужно сначала выяснить детально, чего именно хочет пользователь (задать вопрос и получить ответ). В самой тривиальной реализации этот сценарий может выглядеть как простой выбор задачи из списка. С применением такого подхода в промышленности можно столкнуться уже сегодня. Позвонив в банк или авиакомпанию,

часто можно услышать робота, который предложит нажать разные кнопки в зависимости от цели звонка (”Для проверки баланса нажмите 1, для уточнения статуса заявки нажмите 2” и.т.п.). Этот подход хорош тем, что не требует от системы никакой интеллектуальности и работает очень надежно. Однако занимает много пользовательского времени и зачастую нервирует. В идеальном случае мы хотели бы, получив запрос в виде предложения на натуральном языке, например ”Какая погода сейчас на улице?”, сразу распознать намерение пользователя.

Это одна из подзадач в построении диалоговых систем, на которой хотелось бы сконцентрировать наше внимание в этой работе. Данная задача является довольно актуальной и стоит уже давно. Так что для ее решения было предложено множество методик. Подробнее они будут рассмотрены в разделе ”Обзор литературы”. Большинство методов ориентировано на работу с английским языком. В основном потому, что для английского собрано наибольшее количество данных. Кроме того, английский просто считается языком по умолчанию в научной среде.

В то время как для английского языка достигнуты внушительные результаты, ситуация с другими языками обстоит несколько иначе. Представленные модели в большинстве своем обучены на колоссальном объеме размеченных данных. Сбор и подготовка подобных наборов данных для других языков, вместе с последующим обучением модели, могут занимать существенное время и вычислительные ресурсы. Поэтому простое перенесение достигнутых результатов путем обучения идентичной модели на другом языке не всегда представляется возможным и рациональным.

Тем не менее, почти для всех мировых языков существуют полные словари и простые переводчики. Пользуясь общими знаниями о связи двух языков (словари, параллельные тексты), можно обобщить зна-

ния одной модели на другой язык, не используя размеченных данных для второго языка совсем (либо используя совсем немного). Данный подход в литературе носит название Transfer Learning.

Целью данной работы является построение модели для извлечения намерения из предложения на шведском языке при помощи переноса знаний, накопленных обученной на английском языке моделью. Шведский язык был выбран из-за его семантической схожести с английским. Оба этих языка принадлежат германской языковой группе. Для того, чтобы протестировать зависимость качества работы методов от схожести языков, мы также провели ряд экспериментов для финского языка. Английский с финским находятся в разных языковых семьях и, соответственно, имеют гораздо меньше общего, чем с шведским.

Выработанную методику можно будет использовать для построения моделей приблизительно такой же точности для любого другого языка.

Постановка задачи

Пусть существует множество команд на английском языке E и конечное множество намерений K

$$K = \{k_1, k_2, \ldots, k_n\}$$

Каждой команде из E однозначно соответствует элемент множества K. Соответствие обозначим J_e

$$J_e : E \to K$$

Тренировочные данные состоят из множеств E, K и соответствия J_e. При этом существует также функция-переводчик T, которая каждой команде на английском языке ставит в соответствие команду на шведском языке. Множество команд на шведском языке обозначим S

$$T : E \to S$$

Целью данной работы является получение функции $J_s : S \to K$ сопоставляющей любой команде на шведском языке намерение. При этом должны выполняться два условия.

1. Намерение должно совпадать с намерением перевода шведской команды на английский язык

$$\forall s \in S \quad J_s(s) = J_e(T^{-1}(s)) \tag{1}$$

$$\forall e \in E \quad J_s(T(e)) = J_e(e) \tag{2}$$

2. Функция J_s не должна зависеть от функции T.

Второе условие является определяющим для данной задачи. Если бы мы могли использовать функцию T в J_s, то можно было бы просто определить J_s как в (1) и остановиться на этом. Но это невозможно,

так как перевод (вычисление T^{-1}) – это слишком дорогостоящая операция. Мы бы хотели получить функцию, которая была бы достаточно легко вычислимой для использования в мобильных приложениях. Поэтому вместо прямого перевода команды со шведского языка на английский и последующего применения существующих алгоритмов мы постараемся выделить какие-то ключевые атрибуты команды на шведском языке и использовать их для распознавания намерения.

В силу требования 2 едва ли возможно удовлетворить требование 1 полностью. Вместо этого поставим задачу построить систему, которая на тренировочных данных сможет максимально часто предсказывать намерения пользователя правильно. В качестве метрики качества предсказания будем использовать долю команд в тестовой выборке, по которой система приняла правильное решение.

Обзор литературы

Задача построения диалоговых систем лежит в области автоматического анализа текстовых данных. Одним из главных вопросов применения нейронных сетей в данной области является эффективное представление слов в памяти компьютера. При анализе текстов необходимо заменять слова на вектора как-то отражающие семантический смысл слова.

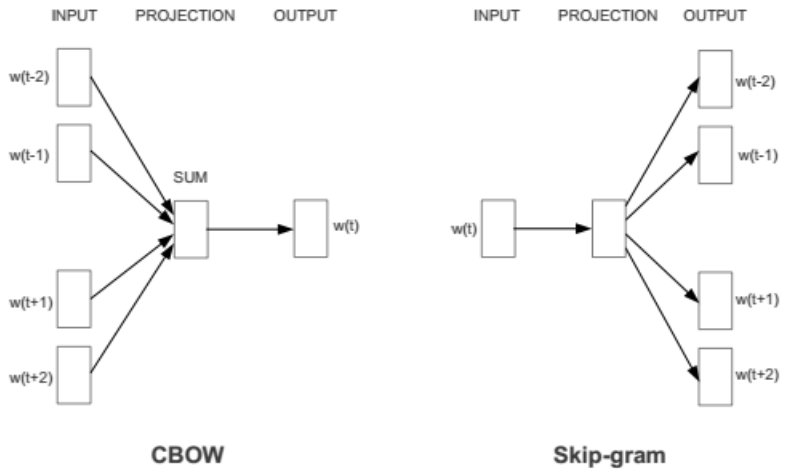
Наиболее распространенный метод получения векторов слов описан в работе [14] В данном подходе обучающий текстовый корпус просматривается окном ширины $(2h + 1)$ слов, и для каждого окна однослойная нейронная сеть предсказывает центральное слово окна w_t по окружающим $w_{t+i}|i \in [-h, h]$ или наоборот. Эти архитектуры называются Continuous Bag-of-words и Skipgram соответственно. Минимизируя ошибку предсказания, нейронная сеть строит проекцию слов в векторное пространство заранее определенной размерности. При достижении заданной точности предсказания или определенного числа эпох, алгоритм генерирует словарь с векторными представлениями для слов из обучающего корпуса.

Данный подход позволяет получить вектора, обладающие свойством семантической близости. Мы можем надеяться, что слова, обладающие схожим смыслом, будут находиться рядом в построенном векторном пространстве. После того как методы описанные в [14] показали свою эффективность в ряде задач обработки текстов [15], было разработано и предложено несколько похожих методов построения векторов слов, таких как ELMo [17] и GloVe [16].

В данный момент большинство методик обработки естественного языка так или иначе использует вектора слов.

В нашей работе мы использовали дополненную реализацию оригинального алгоитма word2vec "fastText" [2]. В отличие от оригинальной

Рис. 1. Архитектуры нейронных сетей, представленные в [14] для окна ширины h = 5



архитектуры, этот подход, помимо полных слов контекста, использует для обучения части слов, что позволяет предсказывать вектора слов для слов, отсутствующих в тренировочной коллекции.

Для работы с последовательностями слов (предложениями) часто применяются рекуррентные нейронные сети, описанные в статьях [18], [11]. Во избежание проблемы затухающих градиентов при обучении используют различные архитектуры рекуррентных модулей, способные лучше "запоминать" информацию. Например, LSTM [10] или GRU[5].

В статье "Attention is all you need" [19] группа исследователей из Google описывает принципиально новый подход к обработке последовательной текстовой информации и, в частности, к переводам. Вместо классической архитектуры рекуррентных нейронных сетей с использованием LSTM или GRU модулей авторы предлагают так называемый механизм внимания, который позволяет более качественно представлять информацию, содержащуюся в предложениях на этапе кодирования. Такую сеть так же называют "Трансформер" из-за гибкости внутренней структуры, позволяющей получать разную инфор-

мацию о кодируемом предложении в зависимости от запроса. Многие современные автоматические переводчики пользуются этой технологией.

В нашей работе мы пользовались готовой системой автоматического перевода от Яндекс. Примерное описание механизмов работы их переводчика доступно в статье [20].

Идея трансформер сетей развивается в статье [7]. Авторы предлагают тренировать многослойную модель из трансформер модулей на задаче определения связности предложений и предсказания пропущенного слова. Полученная модель показывает исключительные результаты после дообучения на ряде конкретных задач.

Задача предсказания намерения из фиксированного множества может быть сформулирована как задача классификации предложений. Интересный подход к этой задаче с использованием сверточных сетей предложен в статье [12].

1. Обработка данных

В нашей работе в качестве тренировочных и тестовых данных мы использовали коллекцию, собранную компанией SNIPS [6]. Данные включают в себя более 13000 запросов пользователей на английском языке. Каждый запрос относится к одной из семи категорий по намерению пользователя.

Представленные намерения:

- Get weather (Посмотреть погоду - 2000 записей)
- Play music (Включить музыку - 2000 записей)
- Book restaurant (Забронировать ресторан - 1973 записей)
- Search creative work (Поиск произведений - 1954 записей)
- Add to playlist (Добавить в плейлист - 1942 записей)
- Rate book (Поставить оценку книге - 1956 записей)
- Search movie schedule (Расписание сеансов кино - 1959 записей)

Данные предоставлены компанией SNIPS по лицензии Creative Commons Zero v1.0 Universal и доступны для скачивания по ссылке `https://github.com/snipsco/nlu-benchmark`.

Таблица 1. Примеры запросов из набора данных SNIPS

Категория	**Пример**
Get weather	What is the forecast at 12 am in Sudan.
Play music	Play some 1954 songs on my Itunes.
Book restaurant	Book four people at a Madagascar bar.
Search creative work	Find me the Lace and Whiskey soundtrack.
Add to playlist	Add this artist to spring music.
Rate book	Give this textbook a 5 out of 6 rating.
Search movie schedule	Where is Road to the Stage playing.

Для подготовки тренировочных и тестовых данных на шведском и финском языке мы использовали API сервиса Яндекс Переводчик[1].

Перед использованием для обучения моделей текст проходил предобработку. Она состояла из нескольких этапов.

1) Приведение всех букв к нижнему регистру;
2) Замена всех пробельных символов на пробелы;
3) Снятие всех ударений, умляутов и подобных знаков. Удаление всех не ascii символов;
4) Раскрытие всех сокращений (Например "You're –> you are");
5) Удаление всех специальных символов при помощи регулярных выражений;
6) Удаление множественных пробелов.

После предобработки набор данных содержал 13784 пары предложений. Медианная длина запроса на английском языке – 8 слов.

[1] `https://translate.yandex.ru/developers`

Самые короткие запросы состоят из двух слов, например: ”play pop”. Самый длинный состоит из 33 слов. Всего предложения включают 11282 уникальных английских слова, каждое слово в среднем встречается 10 раз.

Медианная длина запроса на шведском языке также 8 слов. Уникальных слов на 14% больше, чем в английском – 12852. Весь набор данных включает по 119529 и 115914 слов в английском и шведском языках соответственно.

2. Алгоритм

2.1. Статистический анализ

Для лучшего понимания структуры данных и с целью развития интуиции для отбора моделей, в ходе работы был произведен небольшой статистический анализ набора данных. Некоторые рассмотренные ниже модели опираются на слово как на единицу информации и рассматривают предложения как неупорядоченные наборы слов т.н. "мешки слов". Было решено сравнить отдельные слова по некоторой величине, отражающей ценность этого слова для классификации предложений.

2.1.1. Метод

Для описания методики вычисления полезности слова для классификации введем некоторые обозначения. Пусть язык E содержит N_e уникальных слов. $\{w_1, \ldots, w_{N_e}\}$ – всевозможные уникальные слова языка E. Слова встречаются в предложениях, всего есть D предложений. $\{s_1, \ldots, s_D\}$. Кроме того, каждое предложение относится к тому или иному классу $k \in K = \{k1, \ldots k_n\}$. Рассмотрим два индикатора. Индикатор принадлежности классу:

$$I_k(s,k) = \begin{cases} 1 & \text{если предложение } s \text{ принадлежит классу } k, \\ 0 & \text{если предложение } s \text{ не принадлежит классу } k. \end{cases}$$

И индикатор вхождения слова в предложение:

$$I_s(s,w) = \begin{cases} 1 & \text{если предложение } s \text{ содержит слово } w, \\ 0 & \text{если предложение } s \text{ не содержит слово } w. \end{cases}$$

Тогда введем следующие обозначения: $c_{i,j}$ – количество раз, кото-

рое слово w_i встретилось в предложениях класса k_j:

$$c_{i,j} = \sum_{0<t<D} I_k(s_t, k_j) * I_s(s_t, w_i)$$

$p_{i,j}$ – доля предложений с меткой k_j среди всех предложений со словом w_i:

$$p_{i,j} = \frac{c_{i,j}}{\sum_{0<t<D} I_s(s_t, w_i)}$$

$l_{i,j}$ – доля предложений, в которых встречается слово w_i среди предложений с меткой k_j:

$$l_{i,j} = \frac{c_{i,j}}{\sum_{0<t<D} I_k(s_t, k_j)}$$

Используя введенные обозначения, запишем выражение для вычисления предложенной статистики $U(w_i)$ отражающей условную полезность слова w_i для задачи классификации:

$$U(w_i) = \sum_{0<j<n} [(p_{i,j} - \frac{1}{n})^2 * l_{i,j}] * \frac{n}{n-1}$$

Значение величины $U(w_i)$ стремится к нулю, при приближении распределения слова w_i по классам к случайному.

$$p_{i,j} \to \frac{1}{n} \forall j \in \{1, \ldots, n\} \implies u_i \to 0$$

При этом, чем более вырождено распределение слова по классам и чем больше предложений класса содержат это слово, тем больше будет значение метрики. Если слово w_i встречается исключительно в предложениях класса k_j и при этом каждое предложение класса k_j содержит это слово, то значение метрики достигнет единицы. Это будет означать, что одно это слово позволяет безошибочно указать на принадлежность всех предложений, содержащих его, к конкретному классу k_j.

2.1.2. Анализ

Вычислив условную полезность для классификации для каждого слова на английском и шведском языке, мы построили рейтинг. В таблице приведены 20 "наиболее полезных для классификации в рамках данной задачи" слов и их U величина. Некоторые слова, являющиеся прямым переводом друг друга, выделены цветом. Кроме того, в таблице приведены значения:

$$mplp = \max_{j=1...n} p_{i,j} \text{ (most popular label probability)}$$

$$mpll = l_{i,j}|j = \arg\max_{j=1...n} p_{i,j} \text{ (most popular label load)}$$

$$label = k_j|j = \arg\max_{j=1...n} p_{i,j}$$

Рис. 2. Топ-20 слов по "полезности для классификации" в английском и шведском языках

#	word	label	u	mplp	mpll	#	word	label	u	mplp	mpll
1	add	AddToPlaylist	0.686	0.996	0.809	1	lagg	AddToPlaylist	0.58	0.995	0.683
2	play	PlayMusic	0.629	0.924	0.883	2	boka	BookRestaurant	0.508	0.966	0.642
3	playlist	AddToPlaylist	0.464	0.931	0.64	3	spellista	AddToPlaylist	0.441	0.942	0.592
4	rate	RateBook	0.436	0.997	0.512	4	spela	PlayMusic	0.415	0.782	0.867
5	weather	GetWeather	0.376	0.999	0.44	5	till	AddToPlaylist	0.388	0.775	0.829
6	movie	SearchScreeningEvent	0.36	0.917	0.514	6	bord	BookRestaurant	0.306	0.994	0.361
7	restaurant	BookRestaurant	0.313	0.993	0.372	7	restaurang	BookRestaurant	0.302	0.993	0.358
8	book	BookRestaurant	0.28	0.764	0.622	8	filmer	SearchScreeningEvent	0.277	0.995	0.327
9	be	GetWeather	0.256	0.909	0.373	9	kommer	GetWeather	0.27	0.915	0.388
10	will	GetWeather	0.248	0.924	0.347	10	det	GetWeather	0.252	0.716	0.653
11	points	RateBook	0.244	0.998	0.286	11	poang	RateBook	0.244	1.0	0.284
12	forecast	GetWeather	0.243	1.0	0.283	12	stjarnor	RateBook	0.243	0.986	0.293
13	out	RateBook	0.24	0.915	0.345	13	vader	GetWeather	0.239	0.998	0.28
14	stars	RateBook	0.234	0.976	0.288	14	betygsatt	RateBook	0.213	1.0	0.249
15	playing	SearchScreeningEvent	0.225	0.978	0.276	15	min	AddToPlaylist	0.209	0.766	0.459
16	table	BookRestaurant	0.223	0.992	0.265	16	musik	PlayMusic	0.176	0.912	0.255
17	my	AddToPlaylist	0.222	0.741	0.527	17	film	SearchScreeningEvent	0.161	0.909	0.235
18	give	RateBook	0.208	0.873	0.333	18	priser	RateBook	0.161	0.992	0.192
19	it	GetWeather	0.207	0.896	0.311	19	bli	GetWeather	0.146	0.997	0.172
20	movies	SearchScreeningEvent	0.201	0.992	0.239	20	biograf	SearchScreeningEvent	0.143	0.997	0.168

В первую очередь, по результатам, приведенным в таблице, можно сказать, что предложенная метрика действительно обладает смыслом. Слова, попавшие в таблицу, похожи на слова, имеющие значение для классификации. Например, слово "play"(играть) имеет высокое

значение для распознавания метки "PlayMusic" (включить музыку). Что вполне логично.

Также можно заметить, что слова, имеющие один и тот же смысл, не всегда находятся на одних и тех же позициях в разных языках. Так например слово "book" в английском языке находится ниже, чем один из его переводов "boka" в шведском. Это связано с тем, что в английском языке слово "book" имеет несколько значений и может означать как "забронировать", так и "книга". И потому имеет меньшую полезность для классификации в этих классах. Аналогичная ситуация наблюдается со словом "table" (стол, таблица).

2.2. Модели векторов предложений

Как уже говорилось ранее, задачи обработки языка подразумевают необходимость представления слов и предложений на естественном языке в каком-либо удобном для обработки виде. Как правило, это вектора. В нашей работе мы использовали представление предложений как векторов, то есть функцию J_s искали в виде:

$$J_s(s) = J_s(V_s(s)),$$

где V_s – преобразование предложения на шведском языке к вектору.

$$V_s : S \to \mathbb{R}^\lambda$$

Для того, чтобы изучать перенос знаний модели для английского языка на другой, нужно сначала подобрать хорошую архитектуру, работающую в одном домене.

В нашей работе мы пользовались несколькими разработанными методиками для представления предложений в виде векторов в некотором пространстве.

2.2.1. FastText – среднее

Для самой простой методики построения векторов предложений мы опирались на готовую модель от Facebook – fastText [9]. Это реализация алгоритма CBOW для получения векторов слов с некоторыми дополнениями. Для обучения модели в качестве входных параметров fastText использует не только слова контекста, как в классической реализации [14], но и символьные n-граммы. То есть все подслова определенной длины. Итоговый вектор слова получается как сумма векторов таких n-грамм. Эта методика была описана командой Facebook AI Research в статье [2] и показала свою эффективность в задачах поиска похожих слов и аналогий. Наличие информации о подсловах при обучении модели позволяет использовать ее для получения векторов слов для слов, ранее не встречавшихся в словаре.

В ходе работы мы сами не обучали модель, а использовали опубликованные в открытом доступе предобученные вектора fastText. Эти данные распространяются по лицензии CC BY-SA 3.0, найти их можно по адресу `https://fasttext.cc/docs/en/crawl-vectors.html` . При обучении векторов использовалась архитектура CBOW с n-граммами длины 5, размером окна 5 и размером вектора $\lambda = 300$. Для обучения использовались тексты с Wikipedia и других открытых источников.

Для получения векторов предложений мы брали среднее значение всех векторов слов, полученных из модели. Такой подход не учитывает порядок и взаимное расположение слов, зато работает очень быстро.

2.2.2. FastText – взвешенное среднее

При усреднении всех векторов слов для получения вектора предложения информация, содержащаяся в отдельных словах, зашумляется. Многие слова не несут полезной для классификации смысловой

нагрузки. Хотелось бы отличать такие слова и не использовать их для усреднения.

Мы протестировали еще одну методику получения векторов предложений на основе fastText. Для получения вектора предложения мы вычисляли взвешенное среднее всех слов, входящих в него. В качестве веса использовалась предложенная нами в предыдущей главе величина полезности слова для классификации $U(w)$:

$$V_s(s) = \frac{\sum_{w_i \in s} V_{fastText}(w_i) * U(w_i)}{\sum_{w_i \in s} U(w_i)}$$

При заранее вычисленных величинах $U(w_i) \forall i \in [1 \dots N]$ такой подход также работает быстро, но позволяет снизить количество шума в векторе и тем самым поднять качество классификации.

2.2.3. ELMo – среднее

В качестве альтернативы fastText мы рассмотрели другую модель для получения векторов предложений – ELMo (Embeddings from Language Models)[17]. Эта методика основана на использовании внутренних слоев обученной двунаправленной языковой LSTM сети для предсказания слов. На первом слое двунаправленной сети подаются вектора слов, полученные из сверточной сети на символах. Затем следуют два biLSTM слоя. Итоговый вектор слова получается как линейная комбинация выходных значений на предыдущих трех слоях.

В нашей работе, мы, так же, как и в случае с fastText, не обучали модель самостоятельно, а использовали предобученную. Данные использованные в работе [4] [8] доступны на сайте `https://github.com/HIT-SCIR/ELMoForManyLangs#downloads`. Данная модель была обучена на корпусе в 20 миллионов слов, случайно набранном из Wikipedia и других открытых источников. Размерность получаемого вектора

слова $\lambda = 1024$.

Данная модель естественным образом использует информацию о расположении слов при вычислении векторов, так как включает в себя двунаправленные LSTM модули. Однако вычисления занимают существенно больше времени в сравнении с fastText.

Как и в случае с fastText векторами, в качестве вектора предложения мы брали среднее значение векторов всех его слов.

2.2.4. Universal Sentence Encoder

USE [3] – наиболее современная из трех рассмотренных технологий, предложенная в апреле 2018 года исследовательской группой Google. Построение векторов предложений, рассмотренное в статье, основывается на использовании глубокой нейронной сети для преобразования отдельных векторов слов к вектору предложения. Данная сеть обучается на различных задачах обработки языка, что позволяет достичь существенных результатов в разных областях. Однако получение векторов требует существенных вычислительных затрат.

В нашей работе мы использовали предобученную модель, реализованную с использованием фреймворка TensorFlow [1], размещенную по адресу `https://tfhub.dev/google/universal-sentence-encoder/2`. Модель распространяется по лицензии CC BY 3.0. Размерность получаемых векторов предложений $\lambda = 512$.

Для оценки качества векторов предложений в контексте задачи распознавания намерения пользователя мы обучали однослойную нейронную сеть предсказывать вероятности классов. Ошибка сети считалась с помощью отрицательной логарифмической функции правдоподобия. Для изменения весов использовался метод Adam [13]. Точность полученного классификатора измерялась как процент правильно предсказанных меток для тестовой выборки с использованием кросс-валидации.

2.3. Линейное преобразование

Допустим, у нас есть модель J_e, которая по вектору запроса на английском языке хорошо предсказывает намерение K. Тогда для решения задачи предсказания намерения для второго языка достаточно построить преобразование векторов запроса со второго языка на английский и затем воспользоваться J_e. Заметим, что такое преобразование не будет являться переводом, так как восстановление предложения по его вектору практически невозможно.

В качестве первого тестируемого метода поиска преобразования мы выбрали простую линейную модель. Функция $L(V_s(s))$ искалась в виде $L(V_s(s)) = V_s(s)A$. Для нахождения матрицы преобразования A решалась линейная система уравнений на тренировочном подмножестве данных.

$$V_s A = V_e,$$

где V_s и V_e – матрицы векторов предложений размерности N_{train} строк на λ столбцов, λ_l – длина вектора предложения в представлении V_l, N_{train} – количество примеров обучающей выборки

Так как N_{train}, как правило, сильно больше λ_s, то система получается переопределенной. Под решением мы понимаем такую матрицу A, которая минимизирует евклидово расстояние между столбцами в правой и левой частях выражения. Для нахождения столбцов матрицы A использовался метод наименьших квадратов.

$$A^{(i)} = (V_s^T V_s)^{-1} V_s^T V_e^{(i)}, i \in [1, \lambda_e]$$

В нашем эксперименте для выбора предсказанной метки приложения мы находили расстояние от образа вектора $V_s(s_i)$ до всех векторов $V_e(e_j)$ из тестовой выборки, кроме непосредственного перевода s_i, и в качестве предсказанного класса брали класс предложения e_j, расстояние до вектора которого было минимальным:

$$J_s(s_i) = J_e(\underset{\substack{j=1\dots N_{train} \\ j \neq i}}{\arg\min}(\|L(V_s(s_i)) - V_e(e_j)\|^2))$$

2.4. Нейронная сеть

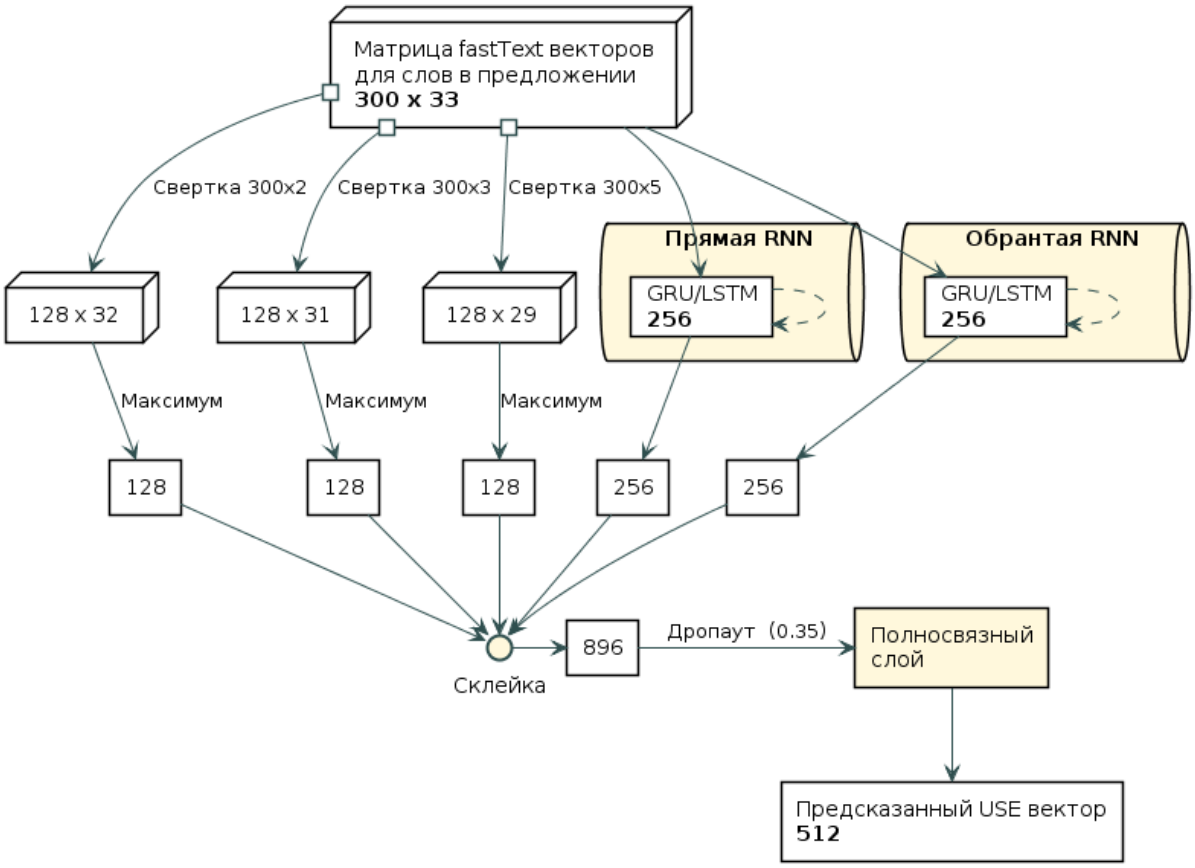
Линейное преобразование – сильно ограниченный в своей гибкости способ построения связи между векторами различных языков. Кроме того, усредняя вектора fastText для шведского языка, мы теряем часть информации. Помимо линейного преобразования, мы также попробовали предсказывать USE вектор предложения на английском при помощи нейронной сети с рекуррентными и сверточными слоями. В качестве входных параметров сеть получала матрицу из отдельных векторов для каждого шведского слова в предложении. Таким образом, алгоритм имел возможность уловить информацию о связи между словами и из взаимном расположении.

Общая архитектура сети выглядела следующим образом. К матрице из векторов слов применялись сверточные фильтры, захватывающие по два, три или пять слов. Параллельно слова по одному подавались на вход рекуррентной нейронной сети с LSTM модулями в прямом и обратном порядке. Два полученных вектора скрытого состояния последнего слоя рекуррентной сети склеивались с векторами из максимальных значений по каждому фильтру. Затем следовал один полносвязный слой. При обучении перед полносвязным слоем 35% случайных элементов занулялись для противодействия переобучению. В результате мы получали вектор нужной нам размерности.

Помимо LSTM модулей, мы также тестировали аналогичную архитектуру с GRU модулями в рекуррентной сети.

Для обучения мы на вход сети подавали матрицу из векторов слов fastText шведского предложения и подбирали веса для минимизации квадрата расстояния между выходом сети и USE вектором соответ-

Рис. 3. Архитектура использованной в работе нейронной сети



ствующего английского предложения.

Для предсказания метки класса мы пробовали два метода.

1) Как и в случае с линейным преобразованием, предсказанным классом шведского предложения считался класс английского предложения из тестовой выборки, чей USE вектор ближе всего к выходу сети.

2) Вектор, полученный рекуррентной сетью, подавался на вход обученному на тренировочных USE векторах линейному классификатору. Метка, предсказанная классификатором, считалась меткой шведского предложения.

3. Результаты

Для проверки гипотезы о том, что качество переноса знаний сильно зависит от семантической схожести языков, мы провели часть экспериментов не только для шведского, но и для финского языка.

3.1. Модели векторов предложений

В данной таблице представлены точности классификации запросов пользователя по классам в зависимости от используемой технологии получения векторов предложений.

Таблица 2. Точность классификации для линейных моделей, основанных на векторах предложений для разных языков

Модель	**Английский**	**Шведский**	**Финский**
FastText-avg	91.9%	88.3%	84.4%
FastText-uw-avg	96.2%	94.2%	94.3%
ELMo-avg	97.7%	96.0%	95.3%
USE	96.8%	–	–

Можно заметить, что предложенный нами метод взвешивания fastText векторов перед усреднением (fastText-uw-avg) позволил существенно улучшить качество классификации. Полученные вектора предложений почти сравнялись по качеству с ELMo-avg векторами.

3.2. Линейное преобразование

В ходе работы было протестировано 6 различных пар представлений предложений на английском и шведском языках. Для каждой пары мы вычисляли матрицу преобразования векторов с шведского

языка на английский, и пользовались ею для предсказания класса шведского предложения.

В таблице представлен средний процент правильно предсказанных меток и среднеквадратичная ошибка после кросс-валидации в зависимости от используемых векторов предложений для разных языков.

Таблица 3. Точность модели с линейным преобразованием векторов предложений

Шведский	**Английский**	**Точность**	**Ошибка**
FastText-avg	ELMo-avg	86.2	0.00745
FastText-avg	USE	89.6%	0.00083
FastText-uw-avg	ELMo-avg	88.8%	0.00881
FastText-uw-avg	USE	91.2%	0.00095
ELMo-avg	ELMo-avg	89.1%	0.00682
ELMo-avg	USE	91.5%	0.00081

Также мы проверили две наиболее интересные пары для финского языка.

Таблица 4. Точность модели с линейным преобразованием fi-en

Финский	**Английский**	**Точность**	**Ошибка**
FastText-avg	USE	88.6%	0.00089
FastText-uw-avg	USE	91.8%	0.00093

3.3. Нейронная сеть

Описанная выше архитектура по векторам слов fastText на шведском языке предсказывала вектора USE для английского. Было про-

тестировано два метода предсказания класса. С использованием "ближайшего соседа", и с использованием USE классификатора. В таблице приведен процент правильно предсказанных меток шведских предложений и среднеквадратичная ошибка между предсказанными и реальными векторами USE.

Таблица 5. Точность модели с нейронной сетью для шведского языка

Рекуррентные модули	**Метод предсказания**	**Точность**	**Ошибка**
GRU	Ближайший сосед	94.2%	0.00068
GRU	USE классификатор	96.4%	0.00068
LSTM	Ближайший сосед	93.7%	0.00069
LSTM	USE классификатор	96.1%	0.00069

Так как сеть с GRU модулями работала незначительно лучше для пары английский-шведский, при тестировании финского языка мы ограничились только этой архитектурой.

Таблица 6. Точность модели с нейронной сетью с GRU модулями для финского языка

Метод предсказания	**Точность**
Ближайший сосед	94.0%
USE классификатор	96.4%

Средний квадрат отклонения предсказанных USE векторов от реальных значений на тестовой выборке составил **0.00069**.

Можно заметить, что полученная архитектура предсказывает метку предложения по fastText векторам лучше, чем рассмотренный ранее линейный классификатор, не использующий знания о связанных английских предложениях и их USE векторах.

Для сравнения мы также пробовали предсказывать метку класса по USE вектору на основании метки ближайшего соседа в пространстве USE векторов без каких-либо преобразований. На всем наборе данных мы получили точность в **94.3%**

Выводы

В данной работе ставилась задача оценить эффективность переноса опыта модели с одного языка на другой на примере задачи предсказания намерения. В ходе работы было построено несколько моделей, использующих этот подход. Протестированные алгоритмы показали точность, сравнимую с точностью простых моделей, работающих на английском языке для шведского и финского. Вопреки ожиданиям, точность для финского языка оказалась на том же уровне, что и для шведского. Это позволяет сделать вывод об эффективности предложенных методов и возможности их использования в реальных задачах для широкого круга языков.

Было замечено что GRU модули в нейронной сети работают незначительно лучше, чем LSTM. Вероятно, более простые архитектуры также позволят достичь сравнимых результатов. Имеет смысл продолжить эксперименты для поиска максимально простой модели, достаточной для полного переноса знаний.

Кроме того, в работе был предложен метод оценки полезности слов для классификации в конкретной задаче. Результаты этого метода были применены для улучшения результатов классификации с использованием одного из алгоритмов.

Заключение

В работе было рассмотрено несколько моделей построения векторов предложений и изучены особенности различных подходов.

Предложен метод ранжирования слов в предложении по полезности для классификации и показано, что такое ранжирование может быть использовано для улучшения некоторых моделей построения векторов предложений.

Было рассмотрено две методики переноса знаний с одного языка на другой: с использованием линейного преобразования и с использованием нейронной сети с рекуррентными и сверточными слоями.

Обе методики были протестированы в разных конфигурациях на разных входных данных. Были получены результаты и показано, что перенос знаний при помощи адаптации векторов слов нейронной сетью позволяет увеличить точность по сравнению с линейной моделью, работающей в одном языке.

Список литературы

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, 2016. USENIX Association.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[4] Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367, 2015.

[6] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre

Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190, 2018.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden, May 2017. Association for Computational Linguistics.

[9] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[11] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.

[12] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning*

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, January 2013.

[15] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting Similarities among Languages for Machine Translation. *ArXiv e-prints*, September 2013.

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[18] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[20] Yandex. Технологии - Машинный перевод, 2019. Дата обращения - 4 апреля 2019.